CS610 Assignment 1
Name: Manish (190477)

SOLUTION 1.

We have given an 8-way Associative cache. Size of Cache = 256kB.
Line Size = 64B and 1 word size = 8B
Size of Array = 32kB
Using the above data we can conclude that:
No. of words in 1 Block = 8 words
No. of blocks: 2^{12}
No. of Sets: 2^{9}
No. of Blocks needed to fit the array = 2^{12}
We can see that our array can completely fit into the cache. There will be no capacity misses and no conflict misses, only cold misses.

No. of Cache Misses:

```
1  double s = 0.0, A[size];
2  int i, it, stride;
3  for (it = 0; it < 100 * stride; it++) {
4    for (i = 0; i < size; i += stride) {
5      s += A[i];
6    }
7  }
```

Case 1: Stride = 1
Out of every 8(No. of words in Block) access, there will be a cold miss. Once the complete array comes into the cache, on further outer loop iterations there will be no misses.
So the number of misses is: 2^{12}

Case 2: Stride = 4
When stride is 4, in every block 2 elements are accessed. Out of which first is cold misses and the other is a hit. On outer loop iterations, there will be no further misses.
So the number of misses is: 2^{12}

Case 3: Stride = 16
When the stride is 16, we only access one element in a block which is a cold miss. On outer loop iterations, there will be no further misses.
So the number of misses is: 2^{11}

Case 4: Stride = 32
When the stride is 32, we only access one element in a block which is a cold miss. On outer loop

iterations, there will be no further misses.
So the number of misses is: 2^{10}

Case 5: Stride = 2k
When the stride is 2k, we only access one element in a block which is a cold miss. On outer loop iterations, there will be no further misses.
So the number of misses is: 2^4

Case 6: Stride = 8k
When the stride is 16, we only access one element in a block which is a cold miss. On outer loop iterations, there will be no further misses.
So the number of misses is: 2^2

Case 7: Stride = 32k
When the stride is 16, we only access one element in a block which is a cold miss. On outer loop iterations, there will be no further misses.
So the number of misses is: 1

SOLUTION 2.

We have a Cache Size of 32k words. Lines of size 8 words. So the number of lines is 4k words. Array dimensions are 512x512(NxN). So the total number of words in the array is 2^{18}. The array is stored in the row-major form.

Loop1(ikj):

Listing 1: ikj form

```
for (i = 0; i < N; i++)
  for (k = 0; k < N; k++)
    for (j = 0; j < N; j++)
      C[i][j] += A[i][k] * B[k][j];
```

Case 1: Direct Mapped Cache
We can see that the $1/8(2^{15}/2^{18})$ array can fit into the cache, which means that index arr(i,j) and arr(i+64,j) will be mapped to the same set.

	A	B	C
i	N	N	N
j	1	N/8	N/8
k	N/8	N	1
Total	$N^2/8$	$N^3/8$	$N^2/8$

Analysis for Array A:
For the innermost loop j, our access pattern is fixed as we are not changing the element we are accessing. So for this loop, the factor will be 1.
For middle loop k, we are accessing Row-wise. Out of every 8 access, 1st is cold miss and 7 are hit. So for this loop, the factor will be $N/8$.
We are accessing new row using the outermost loop i so all are cold misses. So for this loop, the factor will be N.
Total = $N^2/8$

Analysis for Array B:
For the innermost loop j, we are accessing Row-wise. So for this loop, the factor will be $N/8$.
For middle loop k, we are accessing new row every time. So for this loop, the factor will be N.
For the outermost loop i, the factor will be N as the array is not completely in the cache.
Total = $N^3/8$

Analysis for Array C:
For the innermost loop j, our access pattern is Row-wise. So the factor of this loop will be $N/8$.
For the middle loop k, Our access pattern is fixed and 1 row of the array can completely fit into the cache. So for this loop factor will be 1.
We are accessing new row using the outermost loop i so all are cold misses. So the factor of this loop will be N.
Total = $N^2/8$

Case 2: 4 Way Associative Cache
The number of sets in the 4-way associative cache is 2^{10}, each set having 4 blocks.
In this cache structure, arr(i,j) and arr(i+16,j) will be mapped to same set.

	A	B	C
i	N	N	N
j	1	N/8	N/8
k	N/8	N	1
Total	$N^2/8$	$N^3/8$	$N^2/8$

Analysis for Array A:
For the innermost loop j, our access pattern is fixed as we are not changing the element we are accessing. So for this loop, the factor will be 1.
For middle loop k, we are accessing Row-wise. Out of every 8 access, 1st is cold miss and 7 are hit. So for this loop, the factor will be $N/8$.
We are accessing new row using the outermost loop i so all are cold misses. So for this loop, the factor will be N.
Total = $N^2/8$

Analysis for Array B:

For the innermost loop j, we are accessing Row-wise. So for this loop, the factor will be $N/8$.
For middle loop k, we are accessing new row every time. So for this loop, the factor will be N.
For the outermost loop i, the factor will be N as the array is not completely in the cache.
Total = $N^3/8$

Analysis for Array C:
For the innermost loop j, our access pattern is Row-wise. So the factor of this loop will be $N/8$.
For the middle loop k, Our access pattern is fixed and 1 row of the array can completely fit into the cache. So for this loop, the factor will be 1.
We are accessing new row using the outermost loop i so all are cold misses. So the factor of this loop will be N.
Total = $N^2/8$

Loop2(jik):

Listing 2: jik form

```
for (j = 0; j < N; j++)
  for (i = 0; i < N; i++)
    for (k = 0; k < N; k++)
      C[i][j] += A[i][k] * B[k][j];
```

Case 1: Direct Mapped Cache
We can see that the $1/8(2^{15}/2^{18})$ array can fit into the cache, which means that index arr(i,j) and arr(i+64,j) will be mapped to the same set.

	A	B	C
i	N	N	N
j	N	N	N
k	N/8	N	1
Total	$N^3/8$	N^3	N^2

Analysis for Array A:
For the innermost loop k, our access pattern is Row-wise and the cache can hold the complete row. So for this loop, the factor will be $N/8$.
For middle loop i, We are accessing new row every time so all are cold misses. So for this loop, the factor will be N.
The cache is not big enough to hold the array. So for the outermost loop j, the factor will be N.
Total = $N^3/8$

Analysis for Array B:
For the innermost loop k, we are accessing column-wise and our cache cannot fit all rows. So for this loop, the factor will be N.
For the middle loop i, our cache cannot fit the array. So for this loop, the factor will be N.
For the outermost loop j, the factor will be N as the array is not completely in the cache.

Total = N^3

Analysis for Array C:
For the innermost loop k, our access pattern is fixed. So the factor of this loop will be 1.
For the middle loop i, Our access pattern is column-wise. So for this loop, the factor will be N.
for the outermost loop j, Array cannot fit into the cache. So the factor of this loop will be N.
Total = N^2

Case 2: 4-way associative cache
We can see that $1/8(2^{15}/2^{18})$ array can fit into the cache and this is 4 way associative, which means that index arr(i,j) and arr(i+16,j) will be mapped to the same set.
The number of sets in the 4-way associative cache are 2^{10}, each set having 4 blocks.

	A	B	C
i	N	N	N
j	N	N	N
k	N/8	N	1
Total	$N^3/8$	N^3	N^2

Analysis for Array A:
For the innermost loop k, our access pattern is Row-wise and the cache can hold a complete row. So for this loop, the factor will be $N/8$.
For middle loop i, We are accessing new row every time so all are cold misses. So for this loop, the factor will be N.
The cache is not big enough to hold the array. So for the outermost loop j, the factor will be N.
Total = $N^3/8$

Analysis for Array B:
For the innermost loop k, we are accessing column-wise and our cache cannot fit complete rows. So for this loop, the factor will be N.
For the middle loop i, our cache cannot fit the array. So for this loop, the factor will be N.
For the outermost loop j, the factor will be N as the array is not completely in the cache.
Total = N^3

Analysis for Array C:
For the innermost loop k, our access pattern is fixed. So the factor of this loop will be 1.
For the middle loop i, Our access pattern is column-wise. So for this loop, the factor will be N.
for the outermost loop j, Array cannot fit into the cache. So the factor of this loop will be N.
Total = N^2

SOLUTION 3.

We have Direct Mapped Cache, size = 16MB, 64B cache lines, 8B per word.
Array Size = 2^{24} or 4096x4096(NxN). The array is laid out in the row-major form.
From the above info, we can deduce that:
No of words in one line/block: 8
No. of words which can be stored in cache: 2^{21}
No. of set: 2^{18}
From the above analysis, we can say that our cache can store 1/8($2^{21}/2^{24}$) array. As it is a direct mapped cache, we can say that arr(i,j) and arr(i+512,j) will be mapped to the same set.

```
double y[4096], X[4096][4096], A[4096][4096];
for (k = 0; k < 4096; k++)
  for (j = 0; j < 4096; j++)
    for (i = 0; i < 4096; i++)
      y[i] = y[i] + A[i][j] * X[k][j];
```

	A	X
i	N	1
j	N	N/8
k	N	N
Total	N^3	$N^2/8$

Analysis of array A:
For the innermost loop i, our access pattern is column-wise. So for this loop, the factor will be N;
For the middle loop j, Our cache is not big enough to store the array completely. So the factor for this loop is N;
For the outermost loop k, Our cache cannot hold the complete array. So outer iterations need data to be loaded again. So for this loop, the factor will be N.
Total = N^3

Analysis of array X:
For the innermost loop i, our access pattern is fixed. So the factor of this loop will be 1.
For the middle loop j, our access pattern is Row-wise. Out of every 8, 7 will be hit and 1 will be a miss. So the factor of this loop is $N/8$.
For the outermost loop k, We are accessing new row so all are cold misses. So the factor of this loop will be N.
Total = $N^2/8$